毎日新聞ニュースサイトをクラウド化
～AWSだからできた、SIerレスなシステム内製化～

2016.6.3 MAINICHI

毎日新聞社　デジタルメディア局
〒100-8051　東京都千代田区一ツ橋１-１-１

第１部

会社概要・サービス紹介
ＡＷＳ導入背景

プロフィール

会　社	株式会社　毎日新聞社
部　署	デジタルメディア局デジタルビジネスグループ
チーム	マネジメントチーム　兼　デベロップメントチーム
名　前	楢本　隆治（ならもと　りゅうじ）　／　３０代中半
担　当	ディレクター ニュースサイトのサイト構成・デザイン・システム・データ分析を担当
好きなAWSサービス	Amazon Cloud Front

アジェンダ

- 会社概要とサービス紹介
- ニュースサイトのシステムの内製化とクラウド化
- なぜ内製化したのか
- ＡＷＳを採用した理由
- 内製化に向けて行ったこと
- 導入効果
- エピソード
- なぜSIerレスでできたのか？

会社概要・サービス紹介

会社概要

- 会社名
  - 毎日新聞社
- 事業内容
  - 日刊紙の発行
  - デジタルメディア事業
  - 雑誌・書籍の発行
  - スポーツ文化事業
  - その他各種の事業
- 創刊
  - 1872年(明治5年)2月21日
  - 今年創刊１４４年目

毎日新聞は、144年の歴史を持つ新聞社


2020年東京五輪

毎日新聞 夕刊 9月9日(月)

IOC総会 決選大差 夏季56年ぶり イスタンブール降す

GDP年3.8%増 消費増税判断 後押し 4～6月上方修正

被災選手 演説決め手


東京日日新聞 創刊号

前身の東京日日新聞の創刊号

新聞協会賞　編集部門・最多受賞

毎日新聞は、マスコミ・ジャーナリズム界の各賞を数多く受賞しています

受賞 「カ士が八百長メール」のスクープをはじめ大相撲八百長問題を巡る一連の報道

大相撲八百長の問題を連日特報した毎日新聞の紙面

受賞 「3・11 大津波襲来の瞬間」をとらえたスクープ写真

沿岸の防風林を越えて押し寄せる大津波に、一気にのみ込まれる住宅地＝宮城県名取市で2011年3月11日午後3時55分、毎日新聞社ヘリから手塚耕一郎撮影

デジタルメディア局の主な事業紹介

ニュースサイト運営

・大手ポータル
・携帯キャリア
・キュレーションアプリ
・電光掲示板
など様々な媒体に適した形で
コンテンツを提供

法人向けニュース提供

MAINICHI Photo Bank

毎索 マイサク

過去記事写真データベース

デジタルメディア局の主な事業紹介

デジタル毎日

毎日新聞の電子新聞サービス。

デジタルならではの便利な機能満載

デジタルメディア局の主な事業紹介

- **電子新聞サービス「デジタル毎日」**
  - ２０１5年６月から開始
  - 毎日新聞ニュースサイトの記事をすべて読める有料サービス
  - 紙面イメージそのままが閲覧できる「紙面ビューアー」
  - 有料会員向けコンテンツ「経済プレミア」「医療プレミア」
  - イベントやプレゼントなどの有料会員向けご優待サービス
  - 朝昼夕のニュースメールと号外メールの配信
  - ウォール・ストリート・ジャーナルも読める

毎日新聞ニュースサイトの紹介

- 最新ニュース、解説・コラムのほか
  デジタルオリジナルコンテンツも提供。

- 写真特集、動画ニュースも展開

- 記事閲覧はメーター制を導入
  - 記事種は有料とプレミア、無料の３種
    （無料は、社説や災害情報など公共性の高いもの）
  - 無料会員の有料閲覧は、月１０本まで
  - 会員登録なしの有料閲覧は、月５本まで

AWS導入背景

毎日新聞ニュースサイトのシステムの新しい取り組み

- ニュースサイトのシステム開発の内製化
- ＡＷＳのクラウドサービスの導入

システムの内製化の背景

- **ニュースサイトの有料化に向けて、日々サイトをデザインやシステム改善、新規サービスの投入など、ユーザに満足していただけるサービスを提供したい**

  – 新たな取り組みを積極的に行える環境作り
  – 実現するためのシステム開発体制・インフラ作り

<u>クラウドがあったから内製化に踏み切れた</u>

内製化に向けて

- **事業部門で開発体制を構築**
  - 開発リソースの固定化
  - 同じフロアーで意思疎通がとりやすい環境作り
- **開発人員確保**
  - 別セクションにいるSI経験を持ったエンジニアを確保
  - 一時的な人員リソースの確保（派遣）
- **クラウドサービスの利用**
  - クラウドの設計・構築は自分たちで行う
  - 過去のシステムから脱却
  - クラウドで提供されているサービスを組み合わせて構築する
- **無理はしない**
  - 必要なものだけ精査して実装。シンプルな構成。

AWSを採用した背景と理由

- ＡＷＳ構築の経験者がいた
- 小さなサービスで実績を積んでいった
- マネージドサービスの種類が豊富
- 運用時の負荷が少ない
- 他のクラウドに比べ、開発情報や開発者が多い
- 急なアクセスにも対応できる
- AWSのアカウントマネージャーのサポートがあった

内製化とAWS導入に必要だった思考の変革

- **トラブルの責任は、自分たちで取る**
- **ＡＷＳでもトラブルが起きることを前提でシステムを構成**
- **詳細な仕様書は、作らない**
  - プロジェクト管理ツールRedmineでチケットを発行して仕様管理
- **システムリリースが終了ではなく、そこからがスタート**
- **自分たちで考える**
- **無理はしない**

人員確保とAWSを使った開発に向けた体制作り

- **内製化に向けてエンジニアの確保と教育**
  - 社内の別セクションにいるSI経験を持ったエンジニアを確保
  - 全員がAWS経験者ではなくても開発できる仕組みづくり
  - 専門性を増すためにAWSアマゾン認定プログラムを取得させた　（現在3名取得）

システム内製化導入の効果

- **起案からシステム化までのプロセスと時間の短縮**
- **開発費用、運用費用が大幅に削減**
- **根本的な障害が発生しなくなった**
  - 自分たちのできる範囲で、複雑なことをしなかった
  - シンプルなシステム構成
- **障害調査、対応の時間が短縮できた**

計画・実施時のエピソード

- **ＡＷＳの請求がドル払い為、起案時は１ドル８０円台だったが、実施したときは、１ドル１２０円台になっており、予算との差が起きた**

  – システム構成の見直し
  – リザーブドインスタンスの購入による大幅削減
  – マネージドサービスの進化による開発軽減
  – トライアンドエラーによるシステムの最適化
  – ＡＷＳの価格改定

計画・実施時のエピソード

- **社内から○○クラウドの方が安い、○○の方が安定していると指摘を受ける**

  – マネージドサービスの多さによる開発費の削減
  – ＡＷＳ構築の開発者が多いことによるスキル取得の短縮
  – リザーブドインスタンスによる大幅ディスカウント
  – トラブルが起こることを前提としたシステム構成

計画・実施時のエピソード

- **新規サービス投入やサービス改善の敷居が下がった**
  - インフラの準備がすぐできることにより、サービスの投入の敷居がさがった
  - Redmineで機能追加や改修の依頼を出せば、見積もり・稟議、要件定義、詳細設計などのSIerに依頼する手間なしに、改修ができるようになった

なぜSIerレスできたのか？

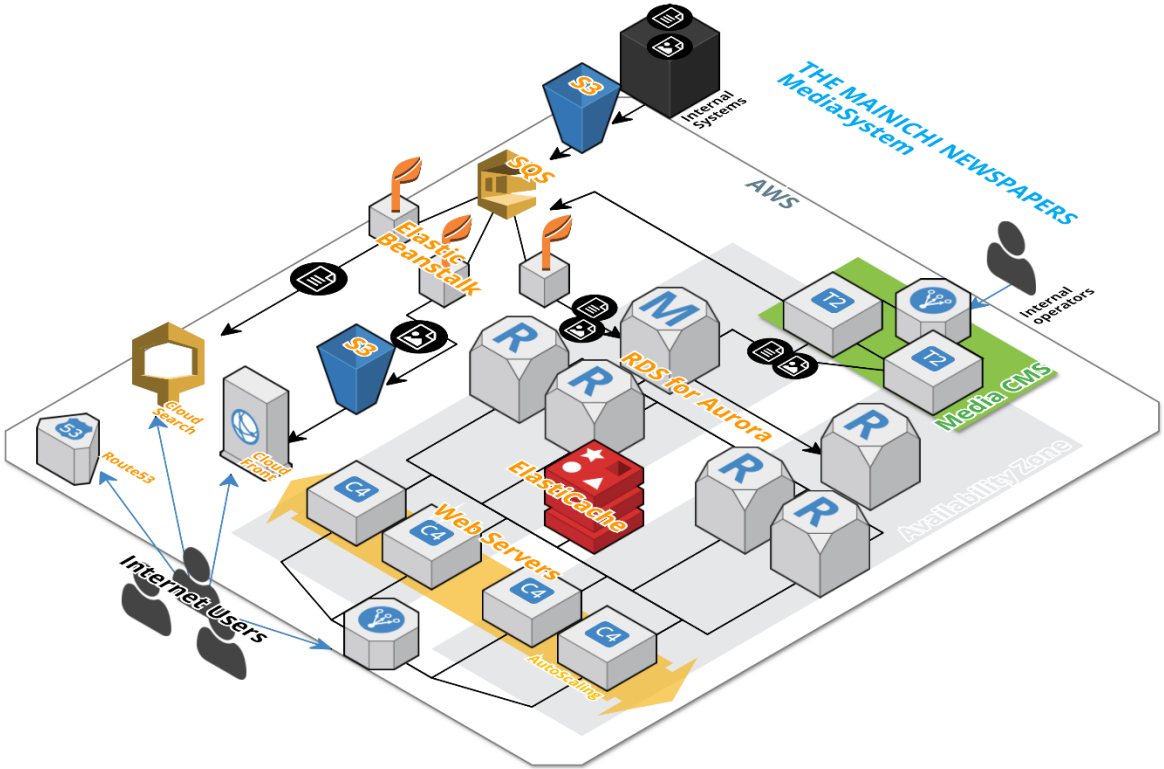
- ＡＷＳのマネージドサービスを利用することにより専門的知識が不要になった
- 物理的サーバの導入・構築・運用の心配が減った
- ＡＷＳを使って、さまざまなサービスをスモールスタートさせて、実績を積んで大規模なサービスに投入
- ＡＷＳの知識がなくても開発できる仕組みを作った
- 無理をしない

第２部
システム紹介

プロフィール

会　社	株式会社　毎日新聞社
部　署	デジタルメディア局デジタルビジネスグループ
チーム	デベロップメントチーム
名　前	森　雄司　（もり　ゆうじ）　／　４０代中半
担　当	システム設計　／　開発現場監督　／　AWS構築及び運用
資　格	**AWS 認定ソリューションアーキテクト – アソシエイト**
好きなAWSサービス	Elastic Load Balancing + Auto Scaling Amazon CloudWatch

システム概略図



POWERED BY CLOUDCRAFT.CO

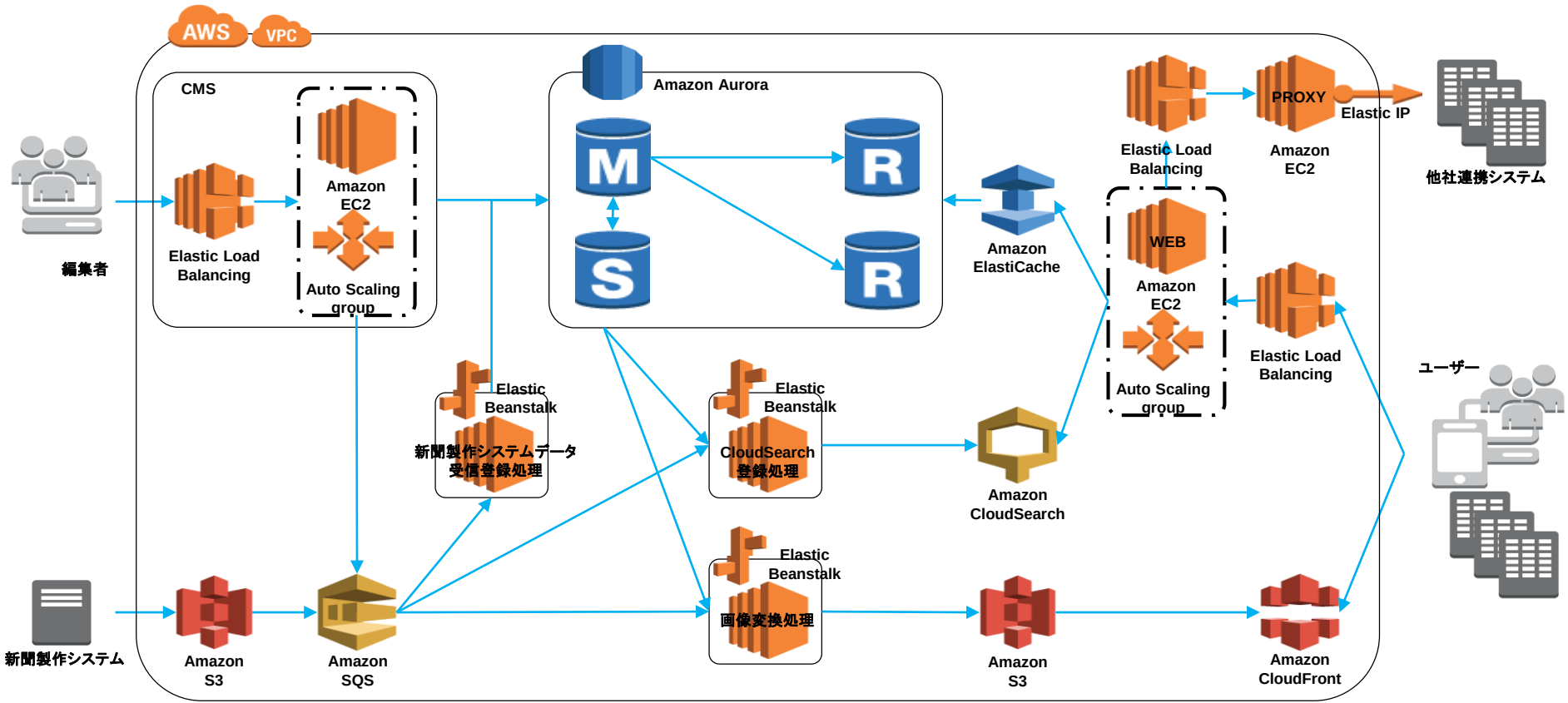
アジェンダ

- システムコンセプト
- システムの構成と機能
- システム監視
- アプリケーション開発の工夫
- Amazon Auroraの導入について
- まとめ

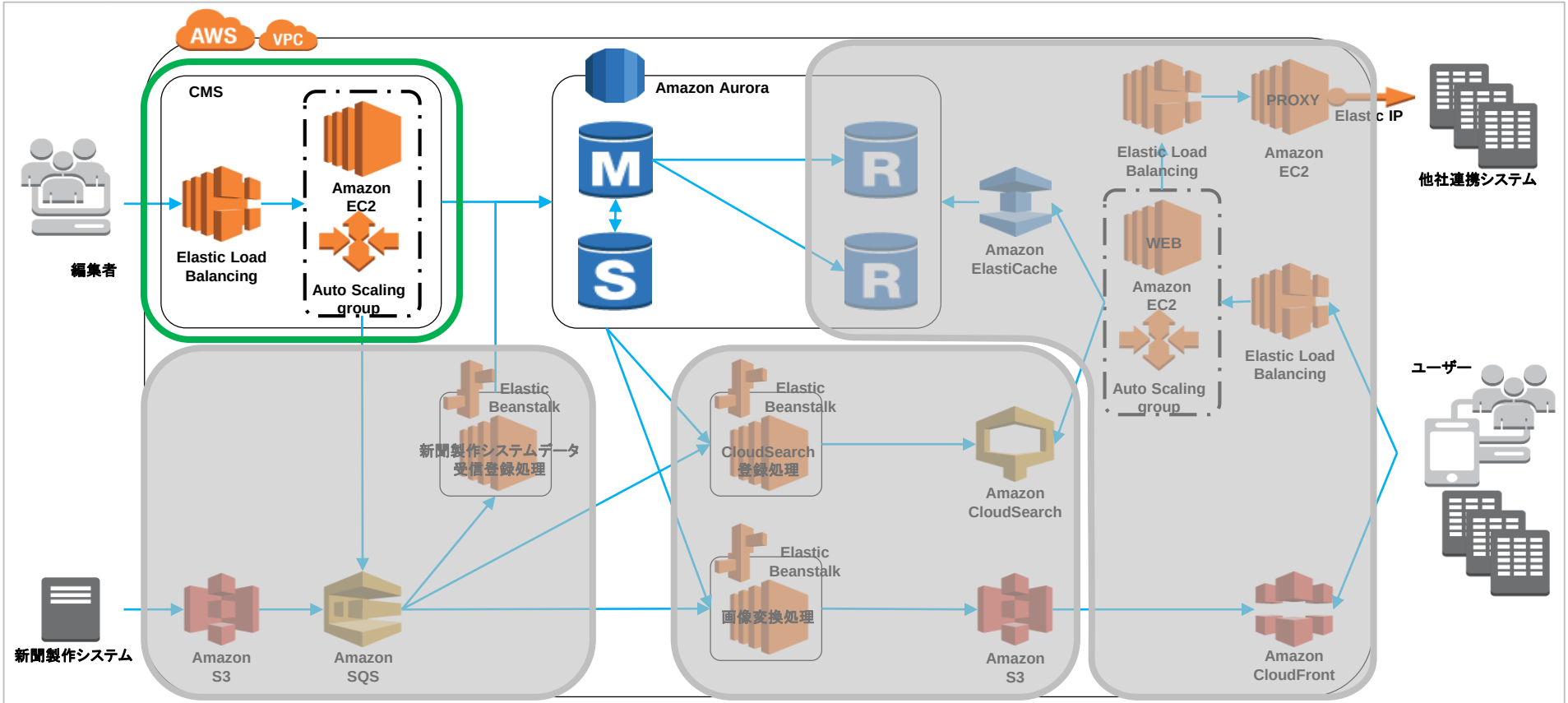
システムコンセプト

- **24時間365日サービスを提供できるシステム**
  - ニュースもライフラインと同じと考え、常にニュースを提供し続けられるシステムの構築
- **オペレータに運用ストレスのないCMS**
  - システム応答速度、メンテナンスによる作業中断が不要など、オペレータがいつでもスムーズに使えるCMSの構築
- **運用保守が容易なシステム**
  - 安定稼働するシステム
  - 保守性・拡張性の高いシステム
  - 障害予知、耐障害性を加味したシステム構成

システム構成図

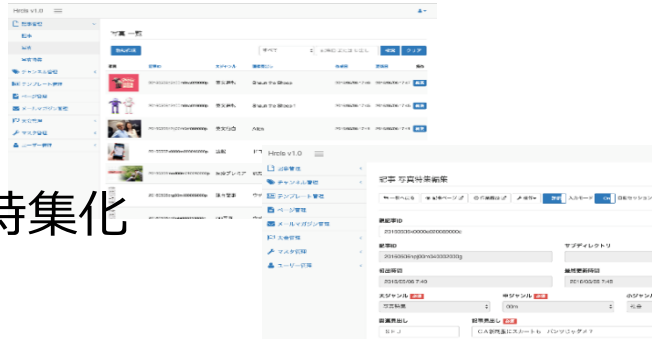


主な機能（1／4）～コンテンツ作成機能（CMS）～



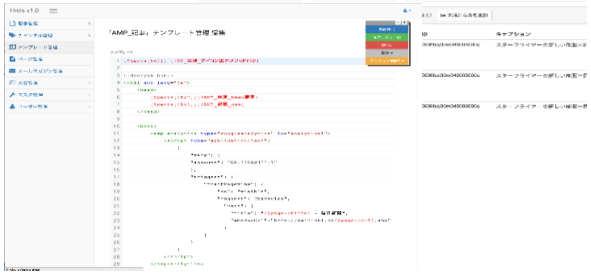
主な機能（1／4）～コンテンツ作成機能（CMS）～

- ニュース作成



- 写真特集作成
  - 複数の写真ファイルを、簡単にまとめて特集化

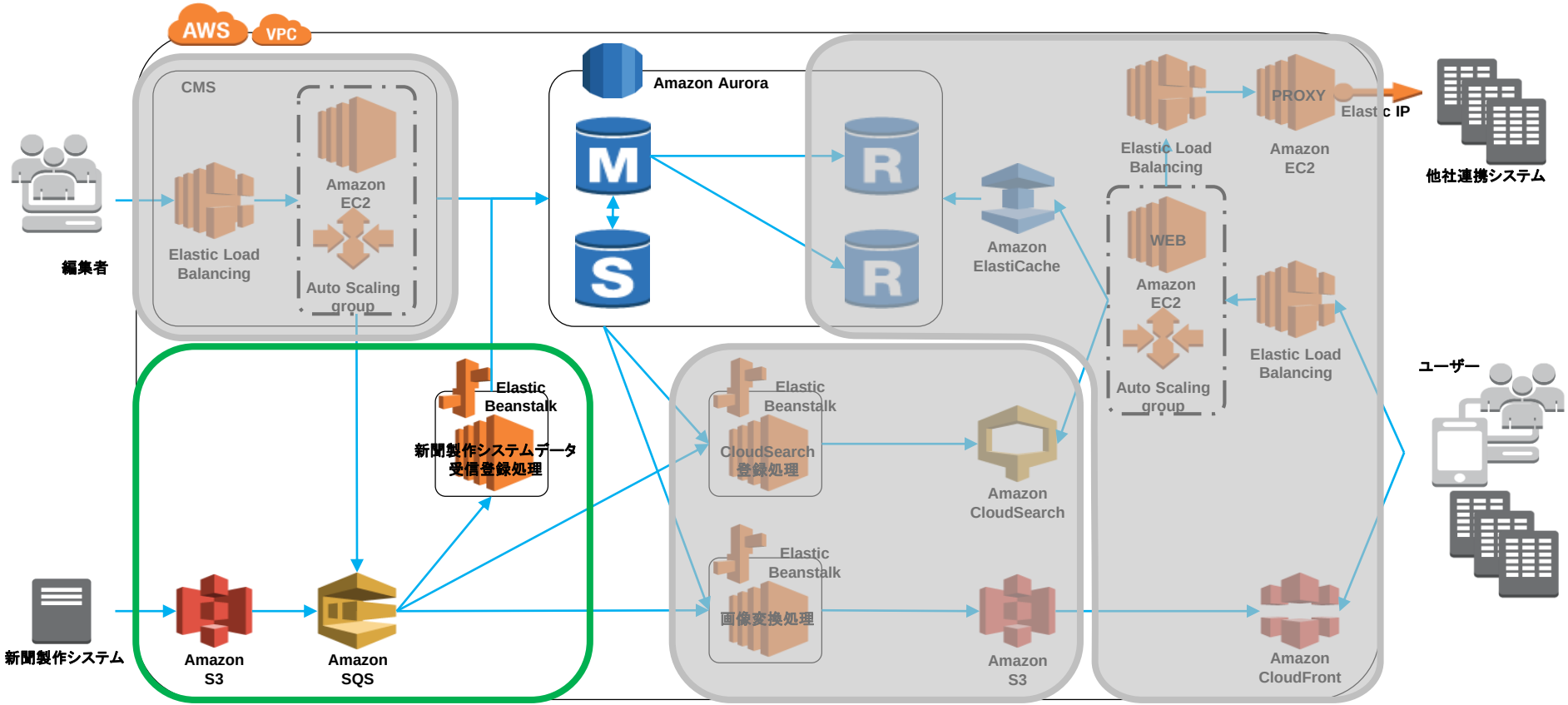
- 特集コーナー作成
  - トレンドに合わせた、コーナーを即時作成



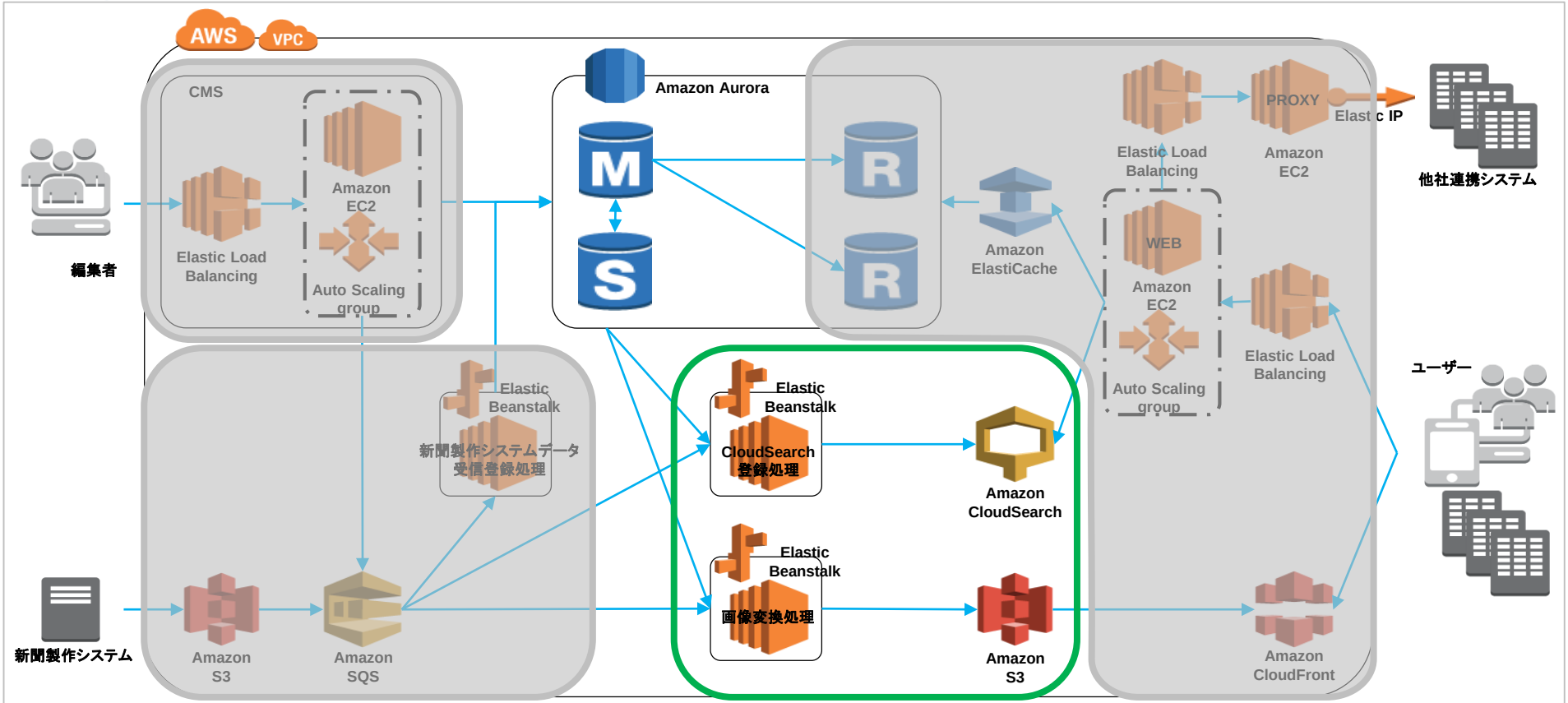
- ニュース及び写真の公開／非公開予約

- テンプレート管理
  - パーツの組み合わせで、様々なページデザインが作成可能

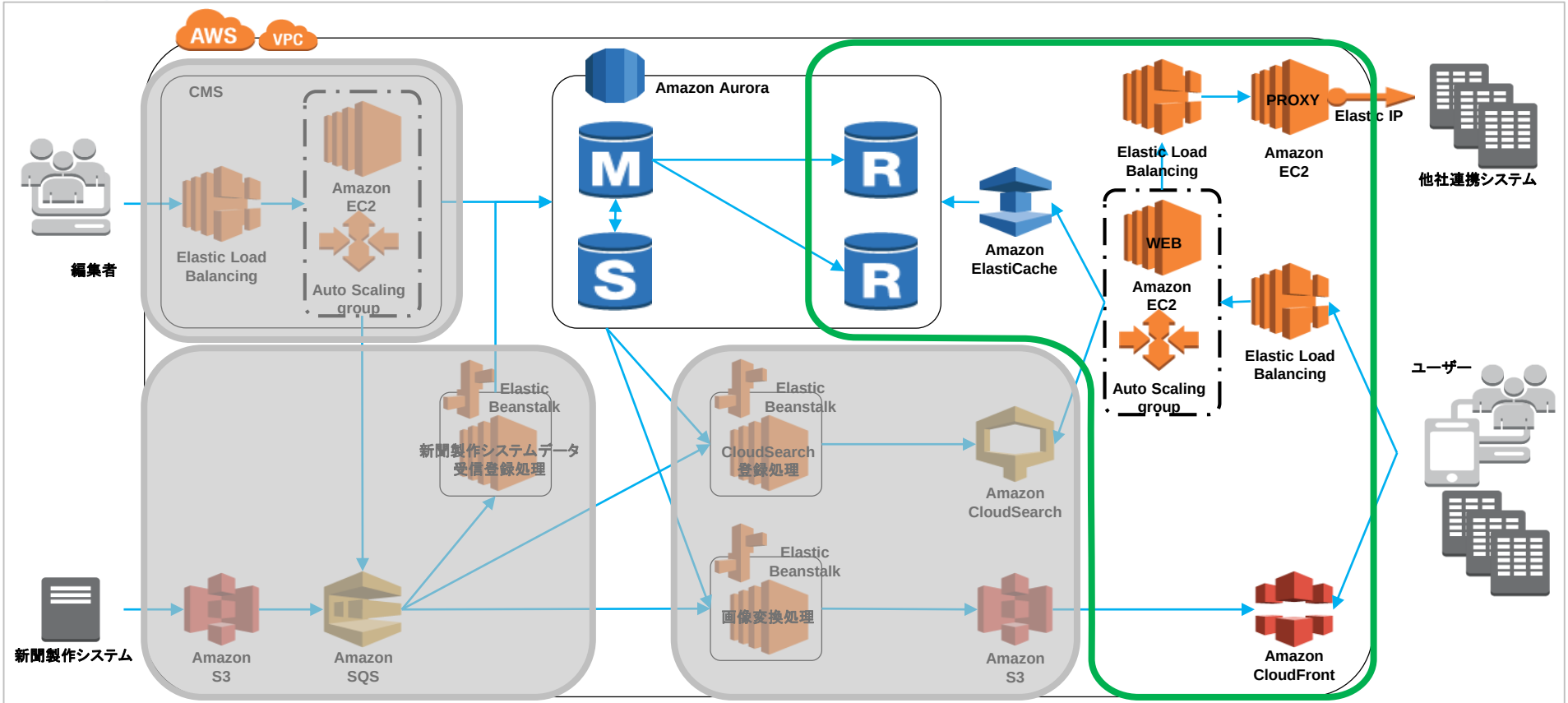
○ 主な機能（2／4）～新聞製作システム連携機能～



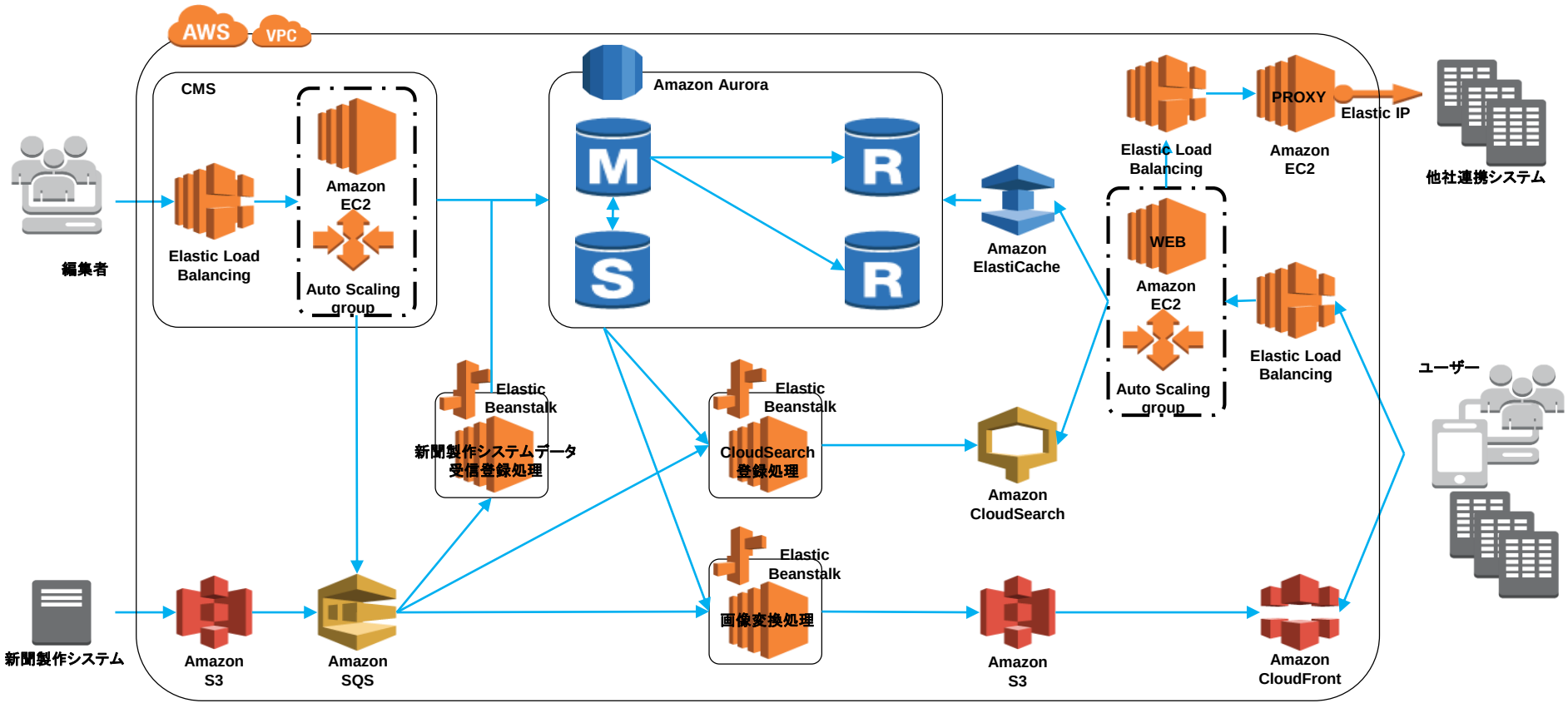
○ 主な機能（3／4）～バックエンド自動処理～



○ 主な機能（4／4）～コンテンツ(ニュース)提供機能～



システム構成図



システム監視

- 「**Amazon CloudWatch**」で稼働状況を監視
  - ダッシュボードに、よく使うメトリックスを登録し、正常時の稼働状況も監視
    - リソースのレビューにも参考にできる

- 「**Amazon CloudWatch Alarm**」＋「**Amazon SNS**」を利用した障害予知
  - アラーム設定でしきい値を超えると、SNSからメールを送信。システム障害が発生する前に対策と対応を行う。

- 「**Amazon CloudWatch Logs**」でログを集約
  - WEBサーバーのアクセスログ、エラーログ、各処理の実行ログを、「Amazon CloudWatch」に集約。「AWSマネジメントコンソール」から容易にログにアクセス。

- 運用監視ソフトウェアによる監視

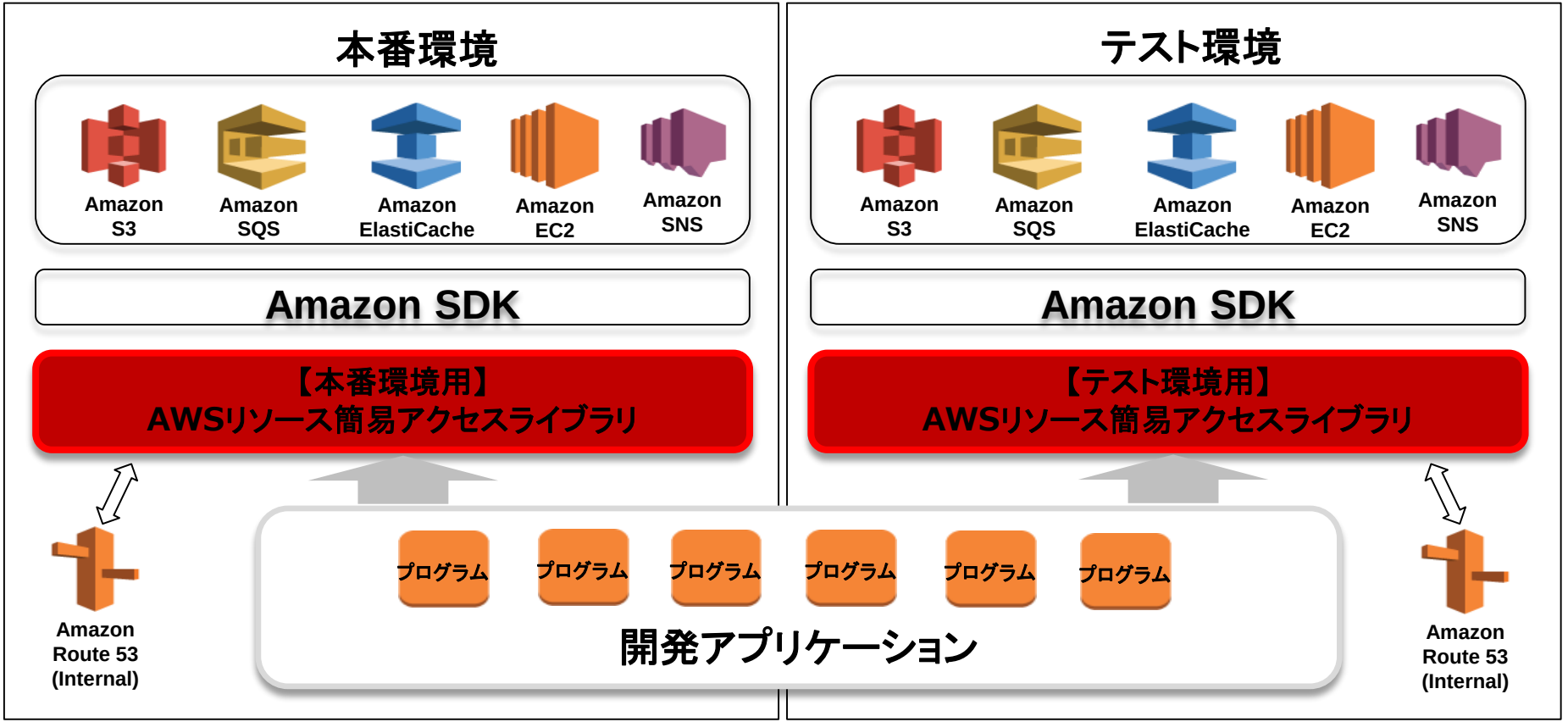
アプリケーション開発の工夫

- AWSリソースアクセス簡易ライブラリの作成
- 処理ごとにリソースを分離

アプリケーション開発の工夫（1／2）

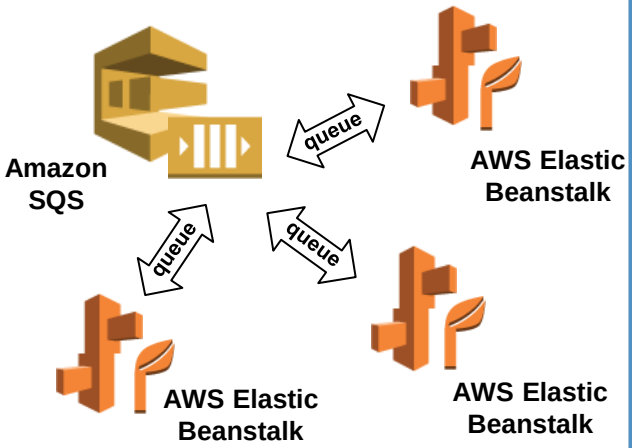
- **AWSリソース簡易アクセスライブラリの作成**
  - 「AWS SDK」から、システムに必要な最小限の機能だけを集めた独自ライブラリを作成
    - メソッドもパラメータも必要最低限
  - 本番環境、テスト環境を意識することなく、プログラミングが可能なため、リリース時の切替えミスが減少
  - AWSの経験がないエンジニアも採用対象とできたため、採用幅を広くできた。

アプリケーション開発の工夫（1／2）

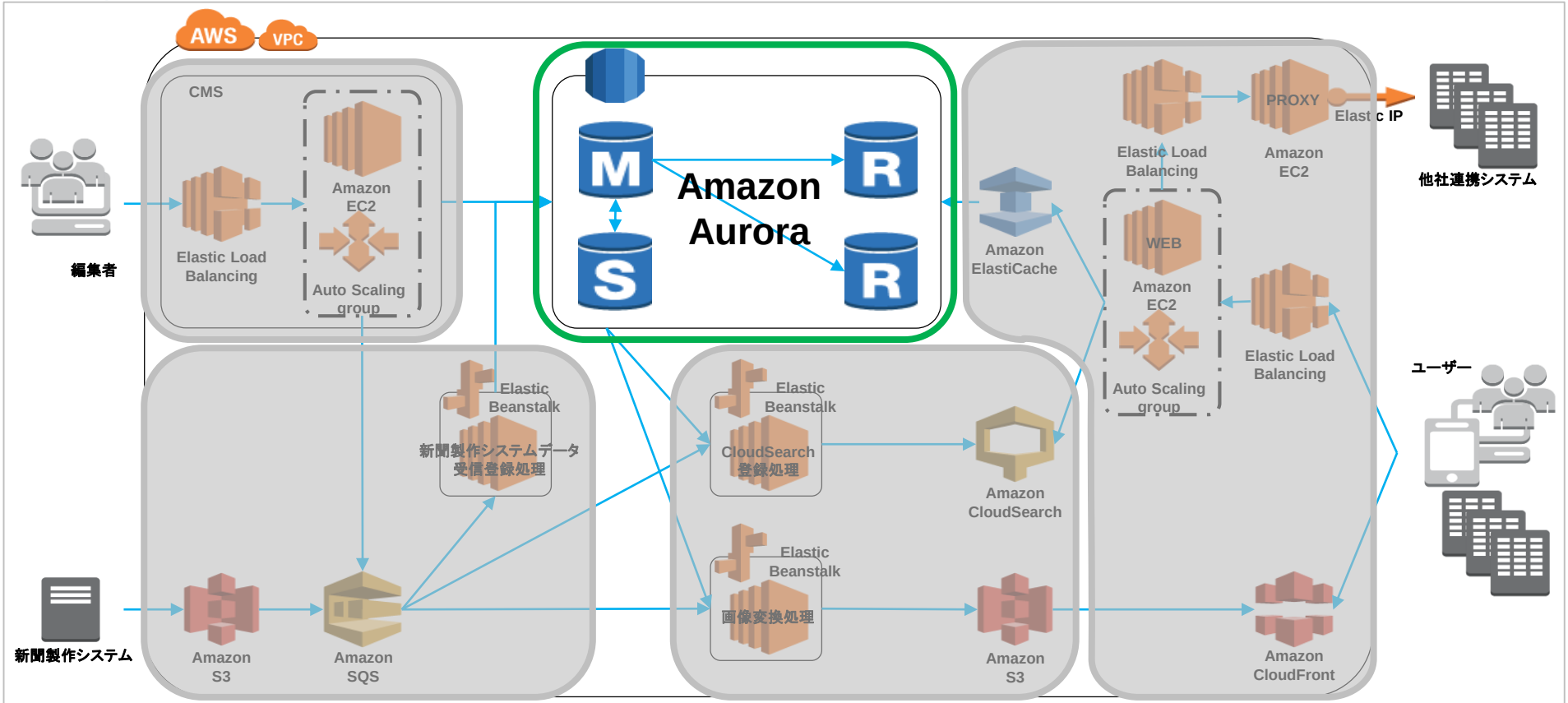


アプリケーション開発の工夫（2／2）

- **処理ごとにリソースを分離**
  - 各処理ごとにリソースを準備。
    - エンジニアは、一つの処理に集中して開発
  - **処理ごとに最適なリソースを割り当て。**
  - プログラムがシンプルになり、プログラムのクオリティーが向上。バグが減った。
  - 障害が発生しても、障害ポイントの特定を早くできる。
  - プログラムのメンテナンスにおける影響範囲が少ない。



Amazon Auroraの導入について

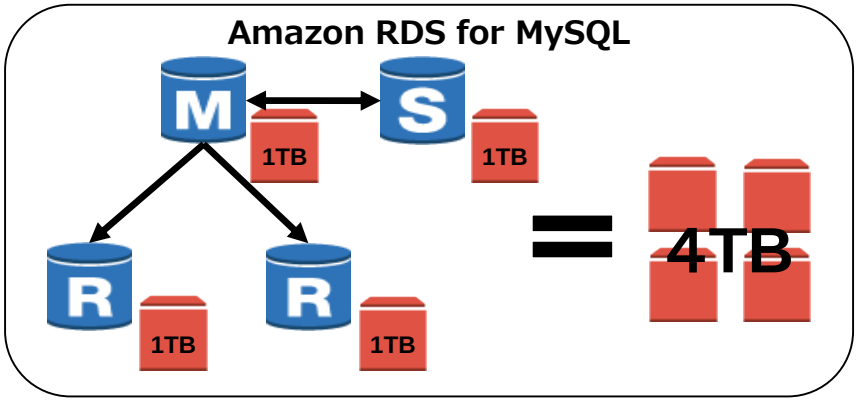


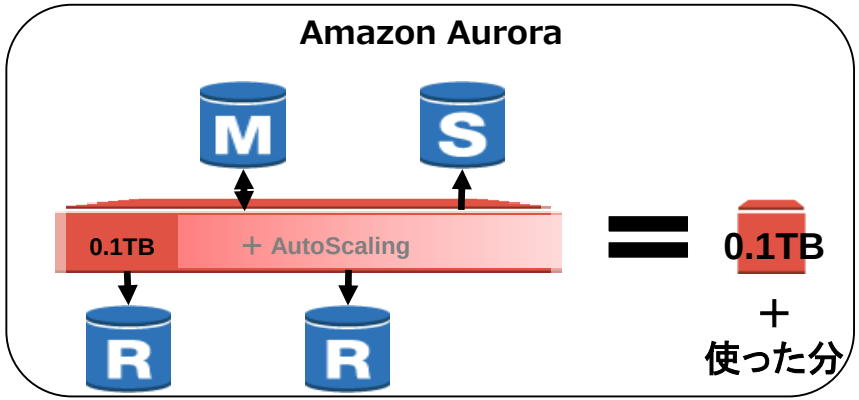
Amazon Auroraの導入について

- 東京リージョンのローンチ　⇒　2015年**10月**
- 当システムのリリース　⇒　2015年**12月**
- 導入を決定　⇒　2015年**11月下旬**
  - 動作検証を行い、導入を決定したのは、**システムリリース2週間前**
  - 設計当初の構成は、「Amazon RDS for MySQL」での「Master + Slave + ReadReplica」の構成

Amazon Auroraの導入について

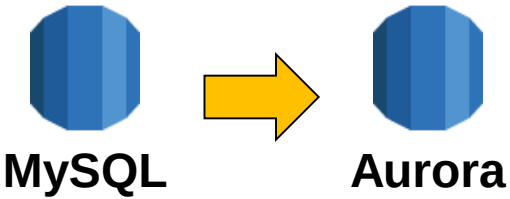
- 導入決断のポイント①
  - **コスト**
    - 設計当初の構成は、「**Amazon RDS for MySQL**」での「Master + Slave + ReadReplica」の構成。この構成だと、最終的に１TBのストレージが必要となる場合は、使用量に関係なく最初から「**Master(1TB) + Slave(1TB) + ReadReplica(1TB) × 2 =4TB**」とインスタンス数分のストレージ費用（月額）が必要となる。
    - 「**Amazon Aurora**」の場合は、インスタンス数に関係なく、**１ストレージ分の費用**で済む。また、容量も最低100GB(デフォルト)から**使っている分だけ(容量はオートスケーリング)**で済むので、最終的にどのくらい必要となるか不透明な、ストレージを購入しておく必要もない。





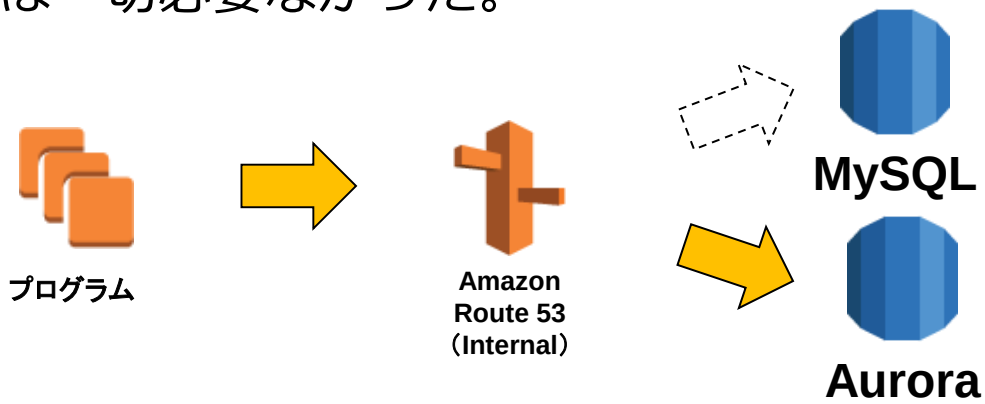
Amazon Auroraの導入について

- 導入決断のポイント②
  - **「Amazon RDS for MySQL」から「Amazon Aurora」への移行が容易**
    - 全てマネジメントコンソールでの操作
    - 「**Amazon RDS for MySQL**」のスナップショットを作成し、そのスナップショットから「**Amazon Aurora**」をローンチするだけ。
    - 所要時間は200GB程度で4時間程度



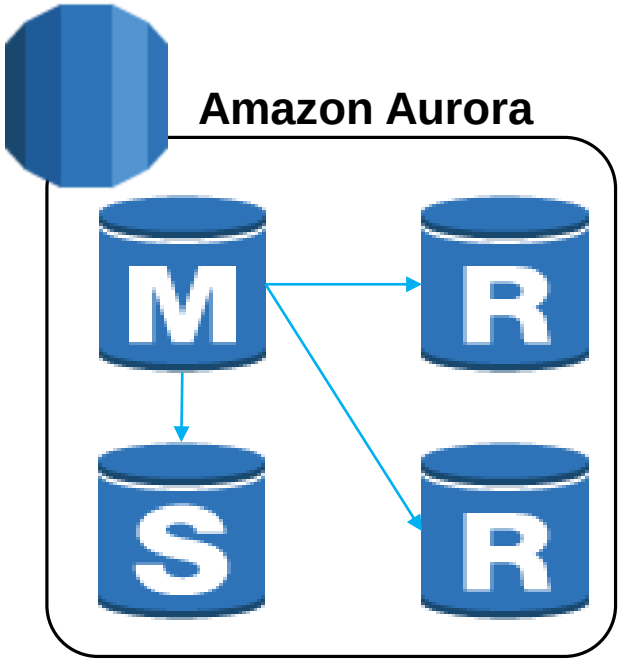
Amazon Auroraの導入について

- 導入決断のポイント③
  - **MySQLとの高い互換性**
    - システムリリース2週間前では、アプリケーション開発も最終段階だったが、「Amazon Route 53」のInternal DNSでエンドポイントを管理していたこともあり、ミドルウェア、プログラム、開発環境の変更は一切必要なかった。



Amazon Auroraの導入について

- 導入決断のポイント（その他）
  - **高パフォーマンスと多機能**
    - 高スループット
    - 99.99%の高可用性
    - 短時間でのフェイルオーバー
    - フェイルオーバー先指定機能
    - 自動バックアップ
    - チューニングレス



備考

- **その他に利用したAWSのサービスや機能**
  - 「**Amazon VPC**」
    - サブネット管理
  - 「**AWS Identity and Access Management（IAM）**」
    - IAM ロールで、各処理で実行可能なオペレーションを管理
  - 「**AWS Certificate Manager**」
    - SSL証明書を無料で利用可能。
    - 「Elastic Load Balancing」への設置も簡単。

今後のシステム展開

- **より障害のないシステムへ**
  - **「Amazon SQS」＋「AWS Lambda」の導入**
    - サーバーレスでより、安定したシステムに。
  - **「AWS CodeCommit」＋「AWS CodeDeploy」＋「AWS CodePipeline」の導入**
    - 安全な開発環境の構築


AWS Lambda


AWS CodeCommit


AWS CodeDeploy


AWS CodePipeline

まとめ

- **安定したアプリケーションの開発**
  - 作る前にリソースを購入するのではなく、**作りながら、リソースを選定**していく
  - インフラに合わせたアプリケーションではなく、**アプリケーションに合わせたリソース**の構築
- **障害を前提としたシステムの構成**
  - メンテナンスや障害が発生して、サービス提供が滞ってしまうポイント、処理負荷が想定されるポイントは、**冗長構成とオートスケーリングで構成**

まとめ

- **マネージドサービスを多用したシステム構成**
  - インフラの構築に必要だった、専門的な知識はほとんど不要
    - 様々な機器の組立てやセッティング　⇒　**不要**
    - ミドルウェアのインストールとチューニング　⇒　**不要**
  - システム保守も容易
    - 物理的なリソースの故障対応　⇒　**不要**
    - マニュアルチックなバックアップ管理　⇒　**不要**
    - サーバー室、電源管理、空調管理　⇒　**不要**

おわりに

ポイント①「安定したアプリケーションの開発」
ポイント②「障害を前提としたシステムの構成」
ポイント③「マネージドサービスを多用したシステム構成」

引き続き、**毎日新聞社**デジタルメディア局といたしましては、AWSのサービスをフル活用し、**内製化**されたシステムだからできる、より良いサービスの提供に心掛けていきたいと思っております。

今後とも、「**毎日新聞ニュースサイト**」及びデジタルコンテンツをよろしくお願いいたします。

ご清聴ありがとうございました。

2016.6.3 MAINICHI

http://mainichi.jp/